Ultracomputer Note #31
TIMING SIMULATIONS FOR ELLIPTIC PDE'S RUN UNDER WASHCLOTH

David Korn
29 June 1981

Chapter I

INTRODUCTION

Two methods for solving three dimensional elliptic partial differential equations have been modified for parallel execution on a paracomputer [9] and run under WASHCLOTH [2,3] simulation.

A brief discussion of each of these methods and timing results under simulation are presented in this note.

CHAPTER 2

CAPACITANCE MATRIX METHOD

A capacitance matrix method for solving the Helmholtz equation on general three-dimensional regions [8] has be modified for parallel execution and run under WASHCLOTH simulation. For more general elliptic problems this method could be used in an iterative fashion.

The region for which the solution is desired is embedded into a rectangular parallelepiped. The algorithm is to alternate between a fast Poisson solver for the parallelepiped and a conjugent gradient method to compute the charges of dipoles at mesh points near the boundary until the given boundary condition is obtained.

For a three dimensional mesh with N points on a side there are O(N**2) boundary points. Since the values for the charges of the dipoles at each of the boundary points is coupled this gives rise to O(N**2) equations in as many unknowns. Fortunately, these equations are strongly diagonally dominant and can be

inverted iteratively by conjugate gradient methods with the number of iterations growing only as log N and the number of operations per iteration proportional to O(N**2).

A Poisson solver using a variant of a Fourier-Toeplitz method [4] has been coded to solve the Helmholtz equation in the parallelepiped. Fourier transformations are applied to two of the three variables and the O(N**2) tridiagonal systems are solved by a Toeplitz method. Since each of the 4N**2 Fourier transforms require O(N*logN) operations the operation count for this portion of the computation is O(N**3*logN) and thus dominates the computing time.

The computer program listed in [8] has been modified for parallel execution and run under WASHCLOTH simulation . While the source program is large (more than 2000 lines of FORTRAN) most of the modifications are straightforward applications of the method described in [6]. code. The computation for each of the O(N**2) boundary points can be carried out simultaneously. All that was required was to change the DO loops to the analogous parallel constructs described in [6]. The floating point replace-add instruction was used in subroutine VMULT since several contributions can be added to the same boundary point.

The Poisson solver was coded so that all of the Fourier transforms could be executed simultaneously in each of the two dimensions. Synchronization is necessary when all the transformations for each dimension are complete and when the tridiagonal systems are complete.

The serial Fourier method is most efficient when several transformations are performed during one call. To maintain this efficiency in the parallel version, a fixed number of transformations were performed at each invocation. It was determined empirically that four transforms taken together achieved better than 90% of the maximum possible efficiency. Lumping four transforms together and performing each of them serially we were able to parallelize the method using up to (N**2)/4 processing elements. Due to the limited memory of the CDC6600 we could only simulate small grids.

Table 1 summarizes the results for an 8x8x8 grid for various numbers of processing elements denoted as #PE. Notice the sharp dropoff of efficiency for 32 processing elements. This occurs because at most 8**2/4 processing elements can be used concurrently when computing the fast Fourier transform. Table 2 summarizes the analogous results for a 16x16x16 grid.

The technique described in [7] for projecting the efficiency as a function of N and #PE can be applied. The waiting time, which is due to serial code, grows linearly with problem size N. The number of instructions that are executed by all PE's is independent of N. The number of divisible instructions grows asymptotically as N**3*logN, the time required for the Fourier transforms.

CAPACITANCE MATRIX METHOD

8 x 8 x 8

#PE	T	TW	Speedup	Efficiency
1	1157K	0	1.00	1.00
2	589K	7K	1.97	.98
4	305K	11K	3.79	.95
8	164K	13K	7.06	.88
16	93K	14K	12.43	.78
32	79K	43K	14.61	.46

Table 1

CAPACITANCE MATRIX METHOD

16 x 16 x 16

#PE	T	TW	Speedup	Efficiency
1	8173K	0	1.00	1.00
2	4102K	12K	1.99	1.00
4	2067K	18K	3.96	.99
8	1049K	21K	7.79	.97
16	540K	23K	15.13	.95
32	285K	22K	28.72	.90
64	157K	22K	52.18	.82
128	142K	71K	57.63	.45

Table 2

CHAPTER 3

MULTIGRID ALTERNATING DIRECTION METHOD

A multigrid method using an alternating direction scheme on each grid was modified for parallel simulation under WASHCLOTH. This method is used to solve elliptic partial differential equations in a parallelepiped with appropriate boundary conditions prescribed. Solutions in more general regions can be obtained by mapping the parallelepiped onto the desired domain providing that the resulting differential equation remains elliptic.

The virtue of a multigrid method is that the number of serial operations needed to solve an elliptic problem is optimal. For a parallelepiped with N grid point on each edge the number of operations is K*N**3 where K depends only on the differential equation and the accuracy required but but not on N. In fact even for modest N the value of K is small enough so that this method is often preferred to methods based on a fast Poisson solver.

The advantage of the alternating direction sub-scheme is that large variances in the aspect ratios of the grid sizes do not slow up convergence. This is especially important when the parallelepiped is mapped to an infinite region.

The idea behind a multigrid scheme is that the rate of convergence for relaxation methods is determined by the errors in the low frequency harmonics of the solution. At fine grids many iterations are needed to remove this error. By using grid sizes that are powers of 2, iterations at each grid size can be used to damp errors at the different wavelengths. Interpolation is used to go from a crude grid to a fine grid and smoothing is used to go from a fine grid to a crude grid.

The number of serial operations is dominated from the work on the finest grid. If we define an iteration to consist of a fixed number of alternating sub-iterations on each of the grids, then the number of iterations required is independent of the grid size. The number of operations per iteration is a sum of the number of operations on each of the logN sub-grids. Since there are N**3/2**j points on each of the j grids, with j from 0 to logN-1, the number of operations is proportional to the N**3 points in the grid.

The alternating direction method requires that we solve (N-1)**2 tridiagonal systems of equations for each of the three dimensions. All of these systems can be solved concurrently. Synchronization is needed after each of the (N-1)**2 tridiagonal systems is solved and when changing grids. The interpolation and smoothing can be simultaneously performed at distinct points in the grid; however, the amount of computation is so small that it seems better to lump N points together and perform only (N-1)**2 tasks concurrently.

Using this method as many as (N-1)**2 simultaneous processes can be used on each of the grids. Since log N grids are used and the time per grid is proportional to N**3/#PE the time to solve this three dimensional system using (N-1)**2 PE's is proportional to N.

A computer program provided by A. Jameson [5] was parallelized and under WASHCLOTH simulation. Table 3 summarizes the results for an 8x8x8 grid. The efficiency loss is primarily due to waiting time. This should be contrasted to the matrix diagonalization code discussed in [7] where the efficiency loss is due primarily to loop initialization overhead performed by each processing element. Table 4 summarizes the results for a 16x16x16 grid.

Projection of efficiency as a function of N and #PE is more difficult for the multigrid scheme. When (N-1)**2 is larger than the number of processing elements and is not an even multiple of it, some of the processing elements must wait for synchronization. This accounts for the decrease in the waiting time per processing element seen in Table 4 when #PE is 64. For the cruder grids fewer processing elements can be used simultaneously. Even though processing elements can not be kept busy on the crude grids, the efficiency can be shown to be greater than .5 even though processing elements are kept busy waiting. If we make the optimistic assumption that multiprogramming can eliminate all the waiting time, the efficiencies rise to the values given in parenthesis in tables 3 and 4.

MULTIGRID ALTERNATING DIRECTION METHOD

8 x 8 x 8

#PE	T	TW	Speedup	Efficiency *
1	2119K	0	1.00	1.00
2	1169K	46K	1.81	.91 (.94)
4	656K	65K	3.23	.81 (.90)
8	400K	75K	5.29	.66 (.81)
16	257K	65K	8.26	.52 (.69)
32	187K	61K	11.34	.35 (.53)

* Values in parenthesis assume no waiting time

Table 3

MULTIGRID ALTERNATING DIRECTION METHOD

16 x 16 x 16

#PE	T	TW	Speedup	Efficiency *
1	18014K	0	1.00	1.00
2	9144K	94K	1.97	.99 (.99)
4	4698K	131K	3.84	.95 (.99)
8	2475K	149K	7.29	.91 (.97)
16	1352K	147K	13.33	.83 (.93)
32	780K	135K	23.10	.72 (.87)
64	476K	112K	37.79	.59 (.77)

* Values in parenthesis assume no waiting time

Table 4

REFERENCES

[1] Achi Brandt, "Multi-level Adaptive Solutions to Boundary-value Problems," NASA-ICASE Report 76-27, 1976.

[2] Allan Gottlieb, "WASHCLOTH - The Logical Successor to Soapsuds", Ultracomputer Note #12, Courant Institute, N.Y.U., 1980.

[3] Allan Gottlieb, "WASHCLOTH 81", Ultracomputer Note #21, Courant Institute, N.Y.U., 1981.

[4] D. Fischer, G. Golub, O. Hald, C. Leiva, and E. Stiefel, "On Fourier-Toeplitz Methods for Separable Elliptic Problems," Math. Comp., Vol. 28, 1974, pp. 349-368.

[5] A. Jameson,

[6] David Korn, "Scientific Code Conversion," Ultracomputer Note #23, Courant Institute, N.Y.U., 1981.

[7] David Korn, "Timing Simulations for Scientific Codes Run Under WASHCLOTH Simulation," Ultracomputer Note #24, Courant Institute, N.Y.U., 1981.

[8] Dianne P. O'Leary and Olof Widlund, "Capacitance Matrix Methods for the Helmholtz Equation on General Three-Dimensional Regions," NYU report COO-3077-155 ,1978.

[9] J. Schwartz, "Ultracomputers," ACM Transactions on Programming Languages and Systems, Vol. 2, pp. 484-521, 1980.